Pattern-Recognition Techniques Applied to Performance Monitoring of the DSS 13 34-Meter Antenna Control Assembly

J. Mellstrom
Ground Antennas and Facilities Engineering Section

P. Smyth
Communications Systems Research Section

This article describes the results of applying pattern-recognition techniques to diagnose fault conditions in the pointing system of one of the Deep Space Network's large antennas, the DSS 13 34-m structure. A previous article described an experiment whereby a neural network technique was used to identify fault classes by using data obtained from a simulation model of the DSN 70-m antenna system. This article describes the extension of these classification techniques to the analysis of real data from the field. The general architecture and philosophy of an autonomous monitoring paradigm is described and classification results are discussed and analyzed in this context. Key features of this approach include a probabilistic time-varying context model, the effective integration of signal processing and system identification techniques with pattern-recognition algorithms, and the ability to calibrate the system given limited amounts of training data. The article reports recognition accuracies in the 97–98-percent range for the particular fault classes included in the experiments.

I. Introduction

This article is intended as an overview of recent research on how to design a health-monitoring system for the DSN antennas. In particular, a systems perspective is provided on how a pattern-recognition component may be embedded within a more standard monitoring architecture. Not included are particular aspects of related technical topics such as classifier design, detection of change in a time series, etc.

The article develops as follows: First, the background to the problem is outlined. This is followed by a general discussion of fault diagnosis, with a description of what has become an evolving design for the autonomous monitoring

system. The conditions for the experiment are described, as well as how the data were collected and why the faults were chosen. The final section is a discussion of the experimental results and the various algorithms that were run.

II. Background and Motivation

The facilities of the DSN, such as the large antenna structures at the Goldstone, Madrid, and Canberra sites, represent a significant capital investment by NASA and JPL. These antenna structures are critical to the performance of the DSN, as they are potential single points of failure in end-to-end network operation. Traditionally, relatively low rates of data loss over the years have occurred due to the efforts of JPL engineering and operations personnel to ensure that problems have been identified and corrected quickly. As long as critical missions such as planetary encounters were of short duration and relatively sparse on the calendar, such labor-intensive approaches were feasible. However, the nature of JPL planetary missions is already changing to involve longer duration planetary encounters such as the Magellan project. In this context, round-the-clock manual supervision of the various antenna structures will not be feasible or economical, hence, the risk of science data loss will be significantly increased. Notwithstanding this change in the DSN operational environment, two other factors independently combine to strongly support the argument in favor of autonomous monitoring of DSN antennas. The first factor is the age of the 70-m antenna systems. The initial projected operational lifetime of these antennas has long been passed, and one can reasonably expect that system components will become more failure-prone as time progresses. The second factor is the planned shift in operational frequencies from X-band (8.45 GHz) to Ka-band (32 GHz); from an antenna-pointing standpoint, this necessitates much more precise pointing accuracies than are currently feasible. In this context, an antenna-monitoring system can, in principle, provide much more information than simply setting off alarms when faults occur. In particular, the monitoring system can complement existing controllers by providing on-line information about pointing-system components.

The above arguments in favor of autonomous health monitoring may be convincing in themselves. However, it should further be noted that recent advances in both hardware and algorithms are what really make an autonomous system feasible. Until recently, the computational capabilities required for real-time data acquisition, signal processing, and pattern recognition for this type of problem would have required large computers at significant cost. A major goal is to produce a system that can be implemented at low cost (perhaps on a single board) using conventional software such as C. This makes for higher reliability and lower cost maintenance from an implementation standpoint. In addition to the recent hardware advances, there has been a resurgence of interest in pattern-recognition techniques and applications for autonomous systems over the last few years, driven largely by the promise of neural network approaches to the problem. From a development point of view this is advantageous, as the significant amount of theoretical and applications-oriented work being carried out in other research institutions can be leveraged.

III. Problem Description: Detecting Failures in Antenna-Control Assemblies

A. Are Antenna Faults a Problem in the DSN?

The importance of the antennas in terms of DSN operation was decribed in the introduction. One question that must be asked, however, is whether antenna subsystems are prone to failure. Direct evidence is difficult to acquire since there is no DSN database that tracks equipment outages at the component level. The easiest information to obtain is that based on discrepancy reports. There are limitations on what can be inferred from these data. In particular, the data outages reported are only a lower bound on the length of time a particular subsystem (and thus a whole antenna) is not available to support tracking. Other inaccuracies can be caused by the reporting mechanism where reports written against the antenna system are due to problems such as wind and snow or procedural errors, which are of no relevance in determining antenna system hardware reliability, although they are relevant for availability.

Nevertheless, one can look at the *relative* magnitudes of data outages caused by the different subsystems in a DSN ground station. Over the period January 1986 to March 1991, 21.9 percent of the total data outage hours (as measured during scheduled tracks) for the DSN as a whole were written against the antenna mechanical subsystem (ANT). The antenna system was the second most likely culprit in this regard after the DSCC Telemetry Subsystem (DTM), which accounted for 28.2 percent of lost data hours, with radio frequency interference (RFI) in third place at 16.3 percent. The top five subsystems account for over 87.7 percent of the total data hours lost. Data outage information is summarized in Fig. 1. In absolute terms, 0.67 percent of all scheduled data hours over the past 5 years was lost because of problems written against the antenna mechanical subsystem. From these data, it seems reasonable to infer that antenna system problems

are a significant contributor to DSN downtime. More detailed information of this nature needs to be acquired, such as outage rates for specific antennas, component-level maintenance information, and so on.

B. Fault Detection and Diagnosis at Present in the DSN

Typically, a faulty component will manifest itself in the antenna-pointing system indirectly via a change in the characteristics of observed sensor points in the control loop. Because of the nonlinearity and feedback present, direct causal relationships between fault conditions and observed symptoms are difficult to establish. This can make manual fault diagnosis a time-consuming and inefficient process. In addition, if a pointing problem occurs while a spacecraft is being tracked, that antenna is often shut down and the track is handed over to another antenna, if possible. Hence, at present, diagnosis often occurs after the fact, where the original fault conditions may be difficult to replicate.

IV. Design of a Fault-Diagnosis System

A. Fault-Diagnosis Techniques

There are a variety of technical approaches for building a fault-diagnosis system. Three such approaches are considered here: control theory, artificial intelligence, and pattern recognition. Willsky [1] described some classic control-theoretic techniques for on-line diagnosis. In general, this approach requires a detailed observer model or filter for each type of postulated fault. This presents a problem, because constructing the observer models is quite difficult due to the high order and nonlinearity of the antenna drive system. In addition, the technique should be adaptive and able to recognize when new faults occur. These capabilities are difficult to incorporate into the conventional observer models. However, the problems are not insurmountable, and the control-theory techniques may offer the most straightforward solution for certain classes of easily identifiable faults.

Artificial-intelligence techniques for fault diagnosis can be classified into two categories. The first is rule-based expert systems. As pointed out in [2], systems with temporal behavior are poorly modeled by the rule-based paradigm. In addition, there is little in the way of experiential knowledge with which to build a rule base for the problem. The second category is that of model-based reasoning, where a high-level qualitative model is built to enable problems to be solved in a manner similar to reasoning from basic principles. While this approach holds promise, especially where novel faults may be expected to occur, the technology is not yet mature enough for application (see [2] for a more detailed discussion).

The third general approach is based on pattern recognition or machine-learning ideas. The idea is simple enough: Design an algorithm that can learn a model directly from observing system behavior. A learning approach precludes the necessity for considerable prior domain knowledge, which is a characteristic of both the other approaches mentioned above. Hence, the system can in some sense be "bootstrapped" into operation. It is a well-known fact in pattern recognition, however, that providing the appropriate domain knowledge to the learning algorithm (say, in the form of preprocessing the data) can considerably improve its performance. Initial investigations bear this point out, and it appears that a hybrid system, which uses components of all three approaches described here, is the most practical and promising avenue. Initial work has been with the pattern-recognition component of the model, which is the primary focus of the remainder of this article.

B. System Design Issues

The approach taken in system health monitoring can be considered to be a rough engineering analogy to passive human sensory and perception capabilities. In particular, there is a hierarchical pattern of information flow, as the raw time-series data are transformed and abstracted by various specialized modules, mapped into a categorical representation of variables of interest (such as fault classes), and finally integrated over time to enable reliable and robust decision making. This architecture is quite generic to systems that must passively sense and perceive their environment in real time, and is a natural choice based on purely engineering considerations.

Figure 2 shows a diagrammatic representation of information flow in the present system. The lower level modules perform dedicated signal-processing tasks to reduce the effective dimensionality of the incoming data. The next level of processing consists of pattern classification operating at a much slower rate in a lower dimensional space. In pattern-recognition terms, there is a feature-extraction stage followed by a probabilistic classifier operating in a continuous real-time mode. Experiments have striven to represent the class information by using probabilities rather than the less informative 1-of-N label information. In this sense, the output of the pattern-recognition component can be viewed as a *probabilistic state vector* that represents the system's best estimate of the state of the system at time t. The decision making component is isolated

to another level of processing after the pattern-recognition component. Note that unlike many applications where decision making and pattern recognition are combined via the use of loss functions, it better serves the purpose here to defer decision making. The separation of probability estimation and decision function allows each component to be modified independently. This is a very important practical design consideration, which allows the decision function to be changed without having to relearn the pattern-recognition component.

V. The DSS 13 34-m Beam-Waveguide Antenna Experimentation Testbed

A. Rationale for Experiment

Having reported initial successes on fault classification by using data from a simulation model [3], the natural next step was to see if similar success could be achieved by using actual data from the field. Obtaining real fault data presents something of a problem, since faults by their nature are unpredictable. Fortunately, with access to the new 34-m beam-waveguide (BWG) antenna at DSS 13, the somewhat bold step of creating faults in the antenna-pointing system in a controlled manner could be taken. One realizes immediately the limitations of this approach:

(1) The procedure cannot readily be repeated on operational antennas, in particular the 70-m antennas.

(2) Only known faults can be simulated, while in a real system, faults of the unknown variety are of great interest.

These limitations are discussed in greater detail later. At this point, it is sufficient to note that despite the limitations, the experiment represents a critical test in demonstrating that pattern-recognition technology can make a real contribution to solving the problems of fault diagnosis.

B. The System Under Study: The 34-m BWG Antenna Elevation Axis Drive Assembly

The antenna elevation axis drive assembly is a closed-loop control system that consists of two 7.5-HP DC motors, servo amplifiers, cycloid gear reducers, tachometers, and electronics for signal conditioning and servo compensation. A simplified block diagram of this system is shown in Fig. 3. A brief description of the system is provided here. Greater detail is provided in [4].

There are two inputs of interest in the antenna drive assembly: the rate command $\dot{\theta}$ and the bias command v_{bias}. In Fig. 3, the rate command $\dot{\theta}$ is applied by the antenna servo controller (ASC). (The ASC is the computer that controls the pointing of the antenna.) The rate command is filtered by the reconstruction filter G_c, then applied to the two subsystems labeled G_o. The term G_o represents two similar sets of amplifiers, motors, gear reducers, tachometers, and compensation electronics, and G_o is itself a closed-loop system referred to as the rate loop. Together, these two subsystems drive the antenna structure.

The remaining block in Fig. 3 is the torque share/bias regulator G_{ct}. This is a regulator circuit that has two functions. The first is to share the torque between the two motors and reduce the effect of parameter variations between them. The second function is to provide a torque bias between the two motors. The torque bias is very important in reducing the nonlinearities and improving gear reducer stiffness. The torque bias removes backlash and shifts the operating point of the cycloid gear reducers into a near-linear, high-stiffness region. The magnitude of the bias is controlled by the bias command.

C. Description of Measured Faults

Five faults were introduced into the antenna drive assembly, all in the drive electronics. The drive electronics were identified as the safest, most easily controlled, and least disruptive location to introduce faults into the drive assembly. Figure 4 shows a block diagram of one motor, amplifier, and gearbox set. Four of the faults were introduced into the rate-loop compensation and tachometer feedback, and the remaining fault was introduced into the torque share/bias feedback loop. During the measurement process, an additional fault appeared in the encoder. Since this was a real fault, it was included as an additional class. The six classes of failure are described below.

(1) The first fault was a noisy tachometer. A uniformly distributed, white-noise process was introduced into the tachometer feedback path of one rate loop, as shown in Fig. 4. This simulates the wear of the commutator or the tachometer bearings, and is a failure that commonly occurs in the DSN and degrades antenna pointing.

(2) The second fault was the total loss of one tachometer. A failure such as this is masked during normal operation because of the interaction of the two rate loops through the torque share/bias loop, and because the closed position loop remains stable.

(3) The third fault was the loss of the torque share/bias feedback to one of the motors. This was implemented by eliminating the signal labeled v_b in one of the summing junctions shown in Fig. 4.

(4) The fourth fault was the elimination of the integrator in the rate-loop compensation, block G_1 in Fig. 4. This effectively changed the rate loop from a Type 1 servo to a Type 0 servo.

(5) The fifth fault was the short circuit of the rate-loop compensation. Referring to Fig. 4, this was equivalent to setting $K_r = 1$, and $\tau_2 = \tau_3 = 0$.

(6) The sixth fault was a real encoder failure. The failure manifests itself as sudden jumps in the measured elevation angle. The magnitude of the jumps was 10 to 20 mdeg. It is hypothesized that the failure is due to binding in the encoder mechanism due to unforeseen radial loading.

Closer analysis of the data revealed that the encoder failure fault was *intermittent* in nature. Detection and classification of this fault is not difficult when using a standard statistical model based on monitoring encoder deviations from the mean expected value. However, this article focuses mainly on the pattern-recognition aspect of the problem. Hence, from this point onwards, only the *persistent* classes of faults are discussed.

D. Data Acquisition at DSS 13

Instrumentation was set up at DSS 13 to measure selected digital and analog signals from the elevation axis antenna drive assembly. These signals were the elevation encoder, current in each DC motor, bias command, rate command, the tachometers for each rate loop, and an average tachometer signal. Also monitored were two anemometers providing wind speed and direction information. Since wind is a nonstationary disturbance, it is important to include this measurement. This provides the environmental context within which nominal performance must be judged. For example, the antenna pointing performance that must be considered nominal subject to a gusty 30-mph wind could be the same as the performance during a calm day with degradation due to a drive fault. On all three days of data acquisition described below, the measured wind speed was less than 5 mph.

Data were collected on three days in January and February 1991. Nominal performance data were collected during antenna calibrations performed on January 25. The data were logged while the antenna was boresighting a radio source. This implies that while the antenna was tracking the radio source, step offsets in elevation and cross-elevation were periodically introduced.

Data were collected under nominal and induced fault conditions on February 11 and 22. The data of February 11 were collected during a simulated high-elevation track. A rate offset of -1 mdeg per sec was introduced. The faults were introduced independently. All the data were collected at elevation angles greater than 60 deg. At elevation angles less than 60 deg, the encoder failure (described above as fault 6) occurred. This precluded the possibility of making any low-elevation measurements.

The data of February 22 were collected while tracking a rising radio source. Again, the faults were introduced independently. Boresighting did not occur during the measurement period. All the data were collected at elevation angles less than 30 deg. Note that all data channels were sampled at 50 Hz to a resolution of 16 bits.

VI. Results of Pattern-Recognition Experiments

The focus of this article is primarily on the pattern-recognition component of the monitoring system design. As outlined earlier, this component is but part of an overall hierarchical strategy for autonomous monitoring.

A. Choice of Pattern-Recognition Technique

Statistical pattern-recognition techniques have been studied since the early 1960s when the availability of computer hardware made it possible to implement computationally demanding classifier design algorithms. A classifier can be thought of as being very similar to a regression equation model, except that instead of trying to predict a real-valued variable, one instead seeks to predict a *categorical* variable, where the categories are the class labels. Typically one has a feature space of K variables; the general idea is to label or partition the K-dimensional feature-space in such a way that an unlabeled feature vector or measurement can be assigned a class label. The standard procedure is to be given a *training set* of labeled sample data, namely N feature vectors or measurements which have class labels attached to them. From the labeled training data, one must *infer* a general relationship between the K features and the class labels: this relationship forms a decision rule or classifier. Let the class labels be c_i, $1 \leq i \leq m$, where m is the number of classes. If the classifier is to return a class label, c_i, without any indication as to confidence of the classifier in this decision, then this is a *discrimination* problem. Denote the K-fold feature vector by the vector random variable $\underline{X}$. If the classifier is to produce the output $p(c_i|\underline{x})$ for each class and any particular feature $\underline{x}$, then this is an *estimation* problem that involves multivariate estimation of the class-feature conditional probability density.

Classifier design techniques for the discrimination and estimation problems fall into two broad categories: parametric and nonparametric. The parametric approach usually involves an assumption that the class-conditional feature densities can be modeled as multivariate Gaussian. Under this assumption the optimal decision boundaries can be found as a function of the means and covariance matrix. The problem with this approach is that Gaussian distributions are often not a good model, and accurate estimation of the components of the covariance matrix requires a large amount of data. The nonparametric approach does not seek a direct parametric form for the conditional densities. Nearest-neighbor techniques, for example, seek to approximate the local value of the density function as a function of the "neighbors" of that point with a given class label. Parzen windows use the same notion of local estimates where the class-probability estimate is based on the weighted contribution of other data points of the same class label, by using various kernel functions. Decision-tree techniques seek the decision boundaries by partitioning the feature space using hyperplanes parallel to the feature axis in a hierarchical manner. All these techniques suffer some drawbacks, such as implementation complexity (nearest neighbor), poor scaling performance as a function of feature dimensionality (Parzen windows), and limited expressive capability (decision trees).

A recent technique that has attracted considerable interest for application to classification problems is that of feed-forward multilayer neural networks. Internal details of such models were outlined in a previous article [3] and some of the details of a particular three-layer network are reproduced in Appendix A. The network implements a set of nonlinear equations that models the relationship between the features and each of the output classes. The network weights play the role of coefficients in the nonlinear equations, and one can view the internal nodes of the network as implementing basis functions. Miller, Goodman, and Smyth [5] have shown how the objective function (used as an error metric to find the optimal set of weights) relates to maximum-likelihood and maximum a posteriori estimation. Similarly, Gish [6] provides a useful discussion about maximum-likelihood properties of network-training algorithms and relates network models to standard multivariate logistic regression models. Among the useful properties of networks are their universal approximation capabilities. Cybenko [7] and Hornik, Stinchcombe, and White [8] have shown that a network with a single hidden layer (a layer of nodes between the input and output) can approximate any continuous function to any desired degree of accuracy. This result is more of theoretical interest than of practical use since it does not prescribe how to find such a network; nonetheless, it demonstrates the important point that network models are a very powerful and flexible basis for approximation.

The problem of network design is then to find a suitable architecture (the number of layers in the model, the type of nonlinearity used) and a good set of weights for the network. The architecture selection problem is still largely done in a fairly ad hoc manner. Typically, a network with a single hidden layer is used with roughly twice as many hidden units (nodes in the hidden layer) as there are classes. Empirical results indicate that network performance is often relatively insensitive to the number of hidden units used (provided that one has a reasonable amount of training data and the number of units is not too small), which indicates a robustness relative to architecture. The number of input units and output units is set equal to the number of features and classes, respectively. Typically, the node "basis functions" are chosen as sigmoid functions (see Appendix A), but any smooth differentiable function can be used as far as the backpropagation training algorithm is concerned.

One of the key contributions to the resurgent interest in neural network models in recent years is the backpropagation algorithm [9], which provides an iterative method for finding a set of network weights that corresponds to a local maximum of the objective function. The algorithm is really nothing more than the application of the chain rule for derivatives applied to a graph structure, coupled with the notion of gradient descent in weight-space. While no tractable techniques are known to exist that guarantee that an optimal or near-optimal solution will be found, practical experience with the algorithm has by and large been remarkably successful. Indeed, while the field of neural networks in general involves a wide variety of biologically inspired computational models, most of the engineering applications of neural networks (and especially the successful ones) hinge on the ability of the backpropagation algorithm to find powerful nonlinear models from data [10–12]. For the experiments described here, an enhancement to the basic gradient descent technique for finding a good set of weights was used, namely, conjugate-gradient optimization, which accelerates the convergence to a solution. More details on this algorithm are presented in [3].

Neural network models offer a powerful new technique for finding classification and prediction models from data. Comparative studies in the literature have consistently shown that network models are as good as, or outperform, other standard pattern-recognition algorithms [13,14], so that network models are now often the model of choice for nonparametric pattern-recognition problems. The significant advantages of a network model are its low complexity

and speed if implemented in hardware, which is important in a real-time application such as this. It is worth noting that multilayer feed-forward network models for classification are finding their way into a myriad of applications in a manner *independent* of the original biological motivations for the model. In other words, as an engineering tool, network models have established themselves in their own right. The application of these models to antenna fault classification is decribed below.

B. Feature Selection and Generation

As described earlier, each data set consists of 12 channels of 16-bit data, sampled at 50 Hz for roughly 5 min for each fault. Hence, for each fault, in pattern-recognition terminology, there are 15,000 12-tuple feature vectors. While this might seem like a large data sample, in actual fact it is really only a very brief snapshot of antenna system data.

Using the raw time-series data directly as input to the classifier is obviously not the best approach to the problem. A hint can be taken from the way a human would discriminate among the classes by using gross structural features of the waveform to characterize it. With this in mind, it was decided that some simple useful features could be extracted directly from the time-series data. (Such "time-domain" features were succesfully used in discriminating fault data from a simulation model of a 70-m antenna [3]). Generation of these features first involved segmenting each channel of the time-series data into windows. The window size was chosen to be of 4-sec duration (200 samples) to give reasonably accurate estimates of the various features. The features consisted of order statistics (such as the range) and moments (such as the variance) of particular sensor channels. A set of seven features was selected which were judged likely to have good predictive power for the problem: motor current range and variance, countertorque range and variance, the range and variance of the average-tachometer sensor, and the variance of the difference between the two tachometer sensors. Even though some of these features are highly correlated, the redundancy is useful to combat any uncorrelated noise that may be present.

The plots in Figs. 5(a) and (b), 6(a) and (b), and 8 all follow the same convention, where the various parameters of interest are plotted versus a window index that corresponds to 4-sec time increments. One can imagine the horizontal axis to be a function of time. The correspondence among the faults and various regions of the axis works as follows: noisy tachometer (1–75), tachometer failure (76–150), bias loss (151–225), integrator short circuit (226–300), compensation short (301–375), and normal (376–450). The entire width of the horizontal axis corresponds to 30 min worth of data.

Figures 5(a) and (b) show plots of the discrimination capability of two of these features on data from "day2" (February 22): the variance of the average tachometer sensor and the motor current variance. Notice that each feature has the ability to discriminate some of the classes and not the others. For example, motor current variance can discriminate the tachometer faults in windows 1 to 150 from the other faults but not from each other, and it can also discriminate the compensation short fault. Notice also that in the variance of the average tachometer sensor there are noise spikes in the data. The motivation for not seeking a completely minimal set of discriminatory features is to retain some redundancy and, hence, robustness in the presence of such noise.

An autoregressive (AR) modeling technique was applied to the motor current sensor. In particular, a variation of autoregression referred to as ARX was used [15], where the "X" refers to an exogenous input variable to the system. The system is modeled by using

$$y(t) + \sum_{i=1}^{p} a_i y(t-i) = \sum_{j=1}^{q} b_j u(t-j) + e(t),$$

$$t = 1, 2, \ldots, N$$

where $y(t)$ is the motor current, $u(t)$ is the system input (in this case, the rate command sensor), $e(t)$ is an additive white-noise process, and a_i and b_i are the model coefficients. A model with $p = 5$ and $q = 3$ was chosen since it provided the best trade-off between goodness-of-fit and model complexity. Note that the more traditional AR model assumes that the system is driven by a white-noise process, i.e., the input $u(t)$ would be replaced by a noise term in an AR model. A 4-sec window was used to segment the data and the ARX coefficients were estimated on a block-by-block basis. For on-line implementation, a standard *recursive* estimation scheme can be used. Figures 6(a) and (b) show the discriminatory capability of the first and second ARX coefficients as plotted for day2 data. Here it is seen that the first coefficient alone is almost sufficient to linearly discriminate among the classes; however, there is some overlap between the tachometer faults, and again between tachometer failure and normal conditions. It is not surprising that the ARX coefficients are useful discriminants since under appropriate conditions they are a sufficient statistic for the data, i.e., they represent all

the information in the data. Consequently, one can expect them to provide more useful information than the simpler time-domain features.

Analysis of the data revealed that the bias loss and integrator short-circuit faults were completely indistinguishable from the normal case. It was expected that some variation would be discernible given these faults; however, it appears that the system is robust enough to withstand such failures. Hence, the data for these two faults were relabeled as normal.

C. Multiple-Network Architecture

Experiments were carried out using a hierarchical set of networks. Specifically, two networks were trained: one on the time domain, the other on the ARX model features. Then the outputs of these two networks were fed into a third "judge" network, which is again trained using the same class labels as the original data. Figure 7 shows the overall model. The intuitive notion behind this multiple-network architecture is that the "judge" network can learn to discriminate which of the first two specialist networks are reliable on which classes. In particular, it learns how to weigh the estimates provided by the first two networks. A useful analogy might be a manager who has some technical experts on staff and needs to weigh their opinions on particular issues, taking care to note the strengths and weaknesses of each. This approach can be seen as a version of the mixture models proposed by Jacobs, Jordan, Nowlan, and Hinton [16]. One can expect this task decomposition approach to work in situations where little gain can be expected from directly combining the features in the two sets. In another sense this can be viewed as one large network where subsets of the feature space are not combined in the early layers to reduce the effect of wasting network resources. This is a more effective and efficient method than simply combining all the features into a single, large, fully connected (between the layers) network, as has been found in numerous large-scale network applications [17–20].

D. Classification Results

As described earlier, the data were collected at DSS 13 on January 25, February 11, and February 22, 1991. This section focuses on the data sets obtained on February 11 and 22, referred to as "day1" and "day2," respectively (no faults were induced on January 25). The goal of the classification experiment was to see if data from each day could be used to predict conditions on the other day. Hence, the methodology can be considered a simple two-way validation test. The results are shown in detail in Table 1. The results for testing on day1 and day2 imply that the models were trained on data from day2 and day1, respectively. Each component network was a three-layer model. The number of hidden units was fixed at 8 for each of the time-domain and ARX networks, at 10 for the network using both sets of features, and the "judge" component of the multiple-network architecture had 8 hidden units.

On average, the multiple network did better than the other models, but the difference is slight. Clearly, most of the useful discrimination ability is in the ARX coefficients rather than the time-domain features, there being an 8-percent difference in mean classification accuracy between these two schemes. These results are quite good if one takes into account that no time correlation is used, i.e., each classification decision is made independently of the other. Clearly, it is preferable to have the scheme correlate its decisions in some manner. In effect, one wishes to model the prior belief that faults are persistent over time and are not likely to change from one 4-sec window to the next. A scheme to do this time-dependent decision making is described in the next section. Note that a popular approach in sequential pattern-recognition problems of this nature is to try and estimate the time dependency from the data. This is usually achieved by providing as inputs to the model (a network in this case) not only feature and class labels at the present time, but also the labels and possibly the features from some window into the past. Such a scheme is viewed as unnecessary in this application since it would make the model much more complicated. Furthermore, the time dependence can be modeled directly. This reinforces the earlier claim that separation of estimation and decision making has beneficial consequences for applications of this nature.

Figure 8 shows a plot of the accuracy of the various models as a function of the sample size. For a fixed sample size k, $50 \leq k \leq 400$; for each model a subsample of size k was randomly selected from the total sample of size 450 from day1. For each k, this sampling was repeated 10 times, the classifiers were trained on the random sample, and their performance was evaluated on the independent data from day2 (all 450 samples). Each point on the graph shows the mean accuracy over 10 such runs. A fixed-size network architecture was maintained for each of the sample sizes since little variation in performance was evident from changing the number of units or layers. The fact that the estimates show little variation as the sample size varies is encouraging, since it shows that the classifier design technique is robust as a function of the amount of training data available. Note also that the time-domain classifier is consistently poorer than the others. Indeed, the ARX model outperforms the single large network for sample sizes greater than 250, indicating that the large

network may be overfitting the noise in the data due to its many degrees of freedom. The multiple-network architecture is consistently better than the other models.

E. Time-Averaged Classification Results

While the ARX model exploits local correlation information at the time-series level, it has already been mentioned that it would be desirable to incorporate correlation information on a more global scale, based on the belief that faults are more likely to persist from one 4-sec window to the next than they are to change.

A relatively simple delayed-decision component was implemented which takes the product of network outputs for each class over the current and some past number of outputs, i.e.,

$$p^n(c_i) = C \prod_{j=0}^{j=M} o_i^{n-j}$$

where $p^n(c_i)$ is the estimated probability of class i at time n, o_i^n is the network output at node i at time n, M is the "memory," and C is a constant independent of the class i. Normalized probability estimates can be obtained by setting

$$\hat{p}^n(c_i) = \frac{p^n(c_i)}{\sum_{k=1}^{m} p^n(c_k)}$$

where m is the number of classes. Appendix B gives a simple analysis of the behavior of this estimator, in particular its robustness to random errors with respect to making classification decisions. Not surprisingly, making the memory M longer reduces the probability of an incorrect decision or false alarm due to random noise effects. Figure 9 shows the smoothing effect of using time-correlated information in this manner on a test data set, using a memory of size 5. In the graph at the bottom, the time-correlation technique results in far less noise in the classification decision than in the other graphs where no memory is used. In the case of the two multiple networks described in the last section (using no information about time correlation), with 8.22-percent and 6.44-percent error rates, the post-processing to take time-correlation into account reduces the error to 2.33 percent and 1.78 percent, respectively.

There is a trade-off in increasing the memory M, in the sense that a longer memory will lead to longer delays in detecting a transition to a new fault. In fact, in Fig. 9, most of the errors occur just after fault-transition boundaries (windows 75, 150, 300, and 375) due to the effective lag in detecting change. More sophisticated decision strategies using context-dependent memory, Markov models, and information from non-neural detectors (such as monitoring the autoregression error sequence to detect change) are also being investigated.

Table 2 shows the actual error rates for the various architectures using a memory of size 5. There is a universal improvement in performance except for one run with the time-domain features, where the smoothing actually made the results worse as compared with no smoothing. The other models achieve error rates between 2 and 3 percent. Table 3 shows the confusion matrix corresponding to the 1.78-percent error rate for the first multiple network in Table 2. Some tachometer failures are misclassified as tachometer noise problems. Also there are some false alarms, two bursts of 8 sec each during the 15 min of normal data. On closer inspection, both of these 8-sec bursts occurred on the "fault" boundaries, when the true class changed from a non-normal to a normal fault, and as such are artifacts of the 20-sec memory scheme. Such false alarms during transitions are not nearly as serious as those that might occur during continuous normal conditions. No such false alarms occurred in the tests described here.

VII. Conclusion

This article describes an application of neural network techniques to pattern classification for a real-world fault-diagnosis task. In particular, the advantages of employing a modular architecture for this problem have been demonstrated, where domain knowledge is brought to bear in designing the lower-level signal-processing modules and the higher-level decision process, and where the task of mapping feature values to class labels is assigned to the neural network model. The initial model proposed was successfully validated on field data.

Acknowledgments

The authors thank D. Ginavan and D. Custer of the Bendix Field Engineering Corporation, and E. C. Posner and R. Stevens of the Office of Telecommunications and Data Acquisition for their assistance in obtaining summaries of Discrepancy Report data.

References

[1] A. S. Willsky, "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, vol. 12, pp. 601–611, 1976.

[2] P. Smyth, "Automated Monitor and Control for Deep Space Network Subsystems," *TDA Progress Report 42-98*, vol. April–June 1989, Jet Propulsion Laboratory, Pasadena, California, pp. 110–120, August 15, 1989.

[3] P. Smyth and J. Mellstrom, "Initial Results on Fault Diagnosis of DSN Antenna Control Assemblies Using Pattern Recognition Techniques," *TDA Progress Report 42-101*, vol. January–March 1990, Jet Propulsion Laboratory, Pasadena, California, pp. 136–151, May 15, 1990.

[4] W. Gawronski and J. Mellstrom, "Elevation Control System Model for the DSS 13 Antenna," *TDA Progress Report 42-105*, vol. January–March 1991, Jet Propulsion Laboratory, Pasadena, California, pp. 83–108, February 15, 1991.

[5] J. Miller, R. Goodman, and P. Smyth, "Objective Functions for Probability Estimation," to appear in *Proceedings of the International Joint Conference on Neural Networks*, Seattle, 1991.

[6] H. Gish, "Maximum Likelihood Training of Neural Networks," to appear in *AI and Statistics 3: Proceedings of the 1991 Workshop*, D. Hand, ed., Chapman and Mall, London, forthcoming.

[7] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing, Vol. 1*, Cambridge, Massachusetts: MIT Press, pp. 318–362, 1986.

[10] D. Touretzky, ed., *Advances in Neural Information Processing, vols. 1, 2, and 3*, San Mateo, California: Morgan Kaufman Publishers, 1989, 1990, and 1991.

[11] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, California: Addison-Wesley, 1991.

[12] R. Hecht-Neilsen, *Neurocomputing*, Reading, Massachusetts: Addison-Wesley, 1990.

[13] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data," *IEEE Trans. Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 540–552, July 1990.

[14] S. M. Weiss and I. Kapouleas, "An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods," *Proceedings of IJCAI 1989*, Palo Alto, California, pp. 781–787, 1989.

[15] L. Ljung, *System Identification—Theory for the User*, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[16] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, no. 1, 1991.

[17] I. Guyon, P. Albrecht, Y. LeCun, J. Denker, and W. Hubbard, "Design of a Neural Network Character Recognizer for a Touch Terminal," *Pattern Recognition*, vol. 24, no. 2, pp. 105–119, 1991.

[18] Y. Mori and K. Joe, "A Large-Scale Neural Network Which Recognizes Handwritten Kanji Characters," *Advances in Neural Information Processing, Vol. 2*, D. Touretzky, ed., San Mateo, California: Morgan Kaufman Publishers, pp. 415–422, 1990.

[19] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard, "Handwritten Digit Recognition: Application of Neural Network Chips and Automatic Learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41–46, November 1989.

[20] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, March 1989.

Table 1. Classification rates on test data sets for classifiers with no time-correlation information

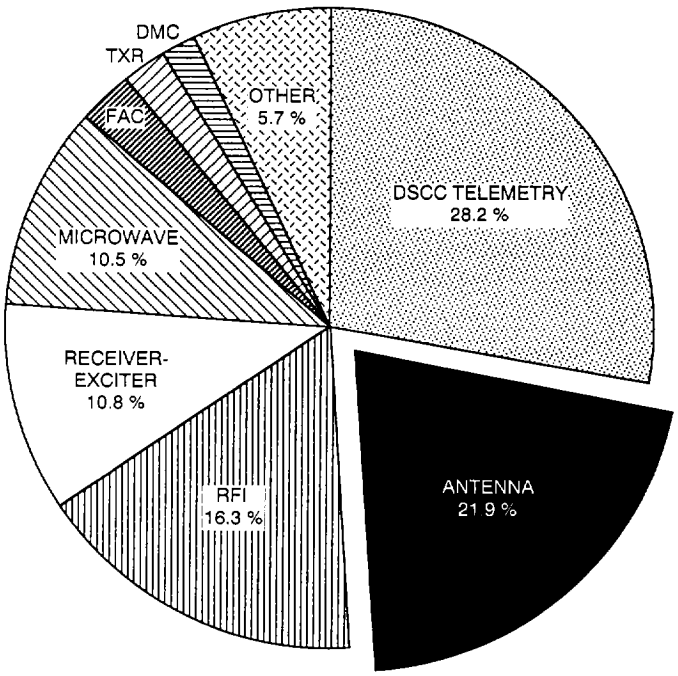
Type of network	Percentage correct		
	Test on day2	Test on day1	Mean
Multiple network	91.78	93.56	92.67
Single large network	90.22	94.00	92.11
ARX model features	91.11	91.78	91.45
Time-domain features	80.00	86.67	83.34

Table 2. Classification rates on test data sets for classifiers using time-correlation information, with memory set to 5 (20 sec)

Type of network	Percentage error		
	Test on day2	Test on day1	Mean
Multiple network	1.78	2.33	2.06
Single large network	3.33	2.33	2.83
ARX model features	3.22	2.33	2.78
Time-domain features	33.44	4.22	18.83

Table 3. Confusion matrix obtained when testing on day1 data set using time-correlation information, with memory set to 5 (20 sec)

True class	Estimated class			
	Tachometer noise	Tachometer failure	Compensation loss	Normal
Tachometer noise	75	0	0	0
Tachometer failure	6	69	0	0
Compensation loss	0	0	73	2
Normal	0	2	2	221



DMC: DSCC MONITOR AND CONTROL
TXR: TRANSMITTER
FAC: DSCC TECHNICAL FACILITIES
RFI: RADIO FREQUENCY INTERFERENCE
DSCC: DEEP SPACE COMMUNICATIONS COMPLEX

Fig. 1. Telemetry data outages during scheduled tracks, January 1, 1986 to July 31, 1990, by DSN subsystem, expressed as a percentage of total hours lost. (Total hours lost was 3.08% of the scheduled support time.)

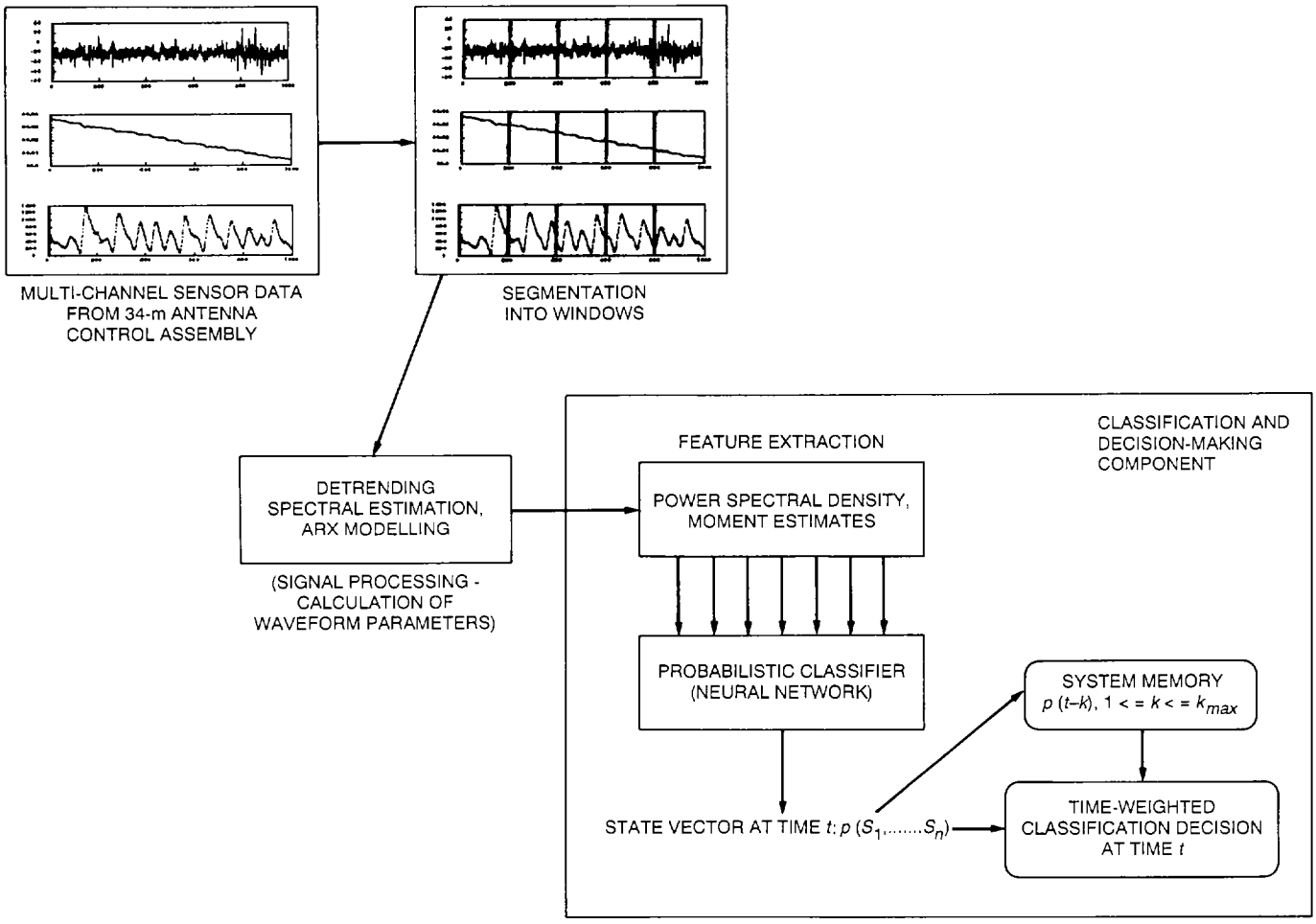


Fig. 2. Hierarchical information flow of the system.

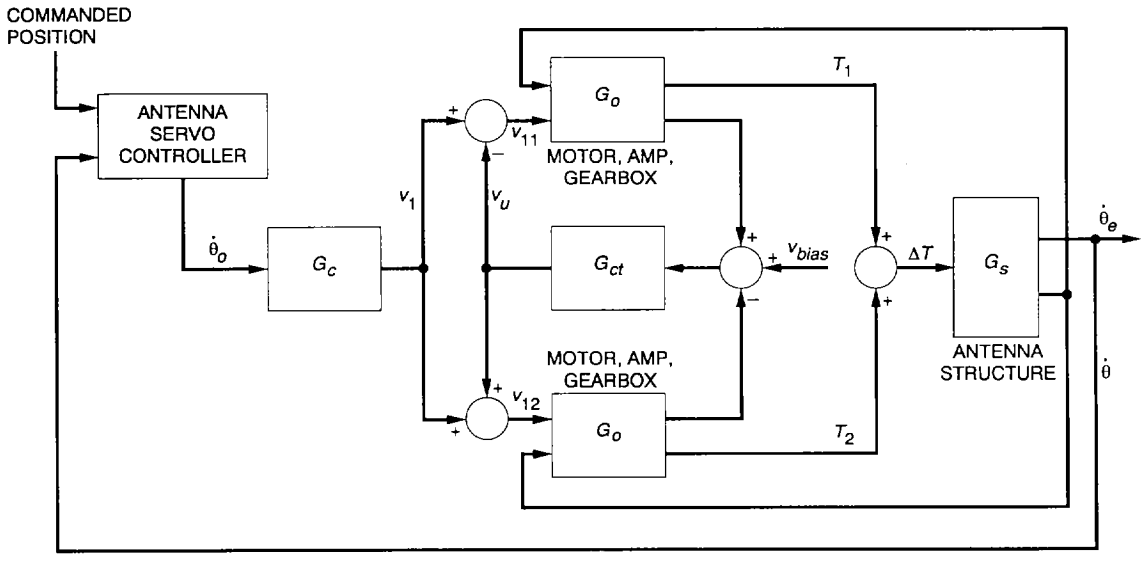


Fig. 3. The 34-m antenna elevation axis drive assembly.

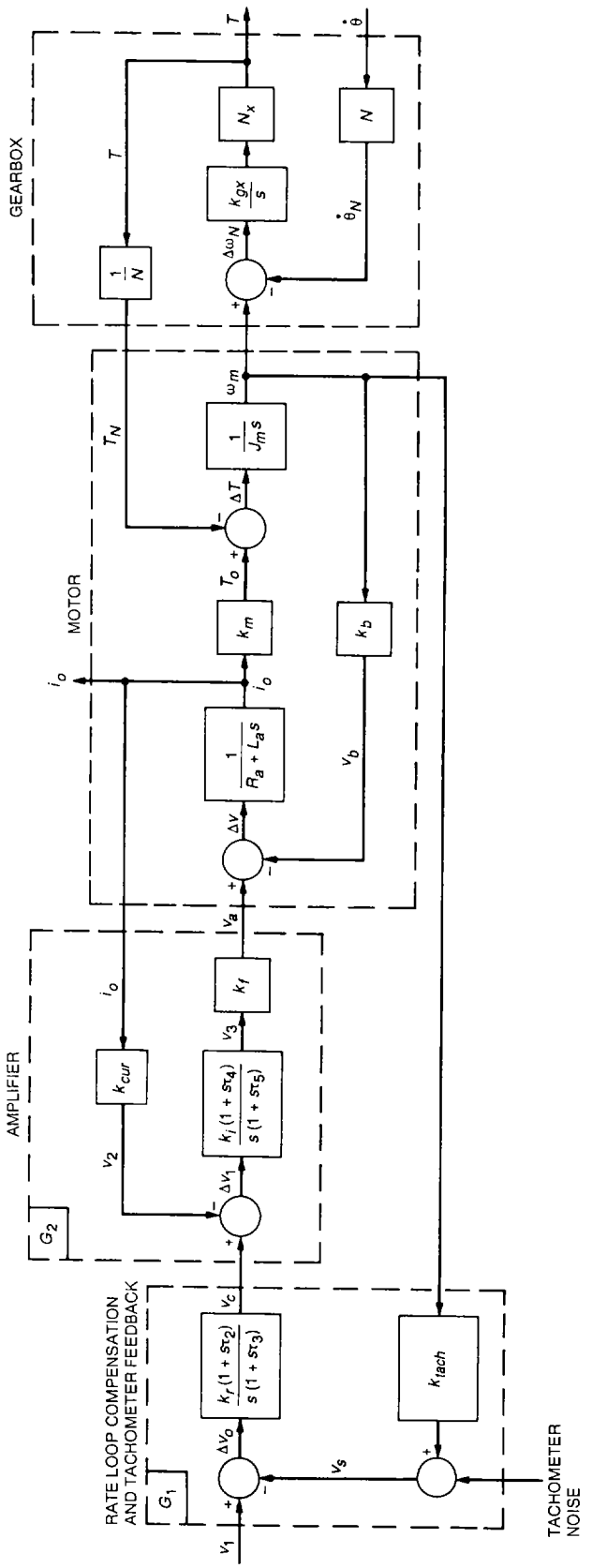


Fig. 4. Motor, amplifier, and gearbox set.

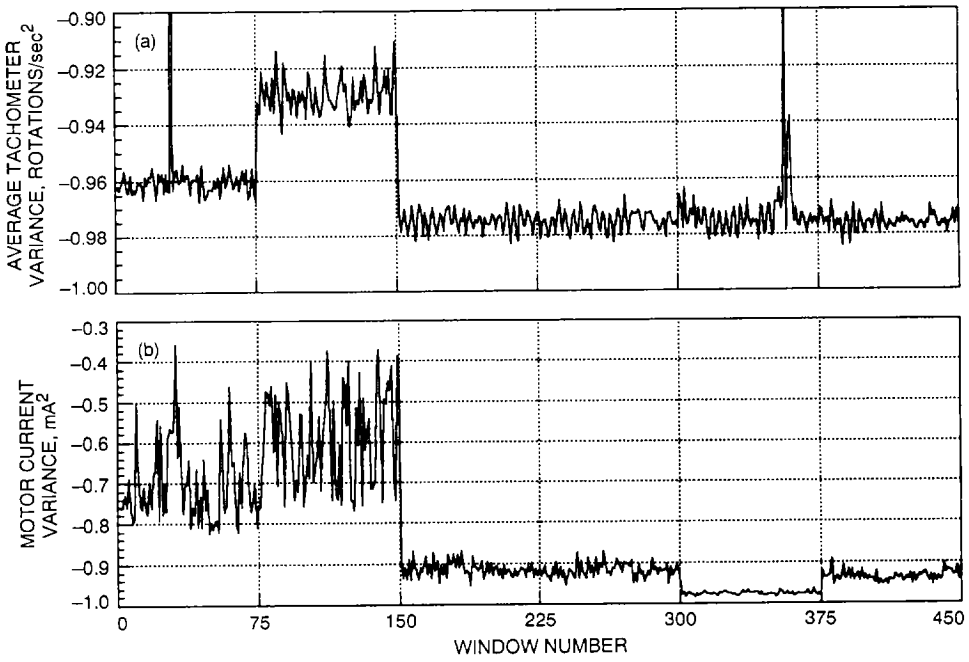


Fig. 5. Discrimination capability of (a) variance of the average tachometer sensor, and (b) motor current variance (day2 data).

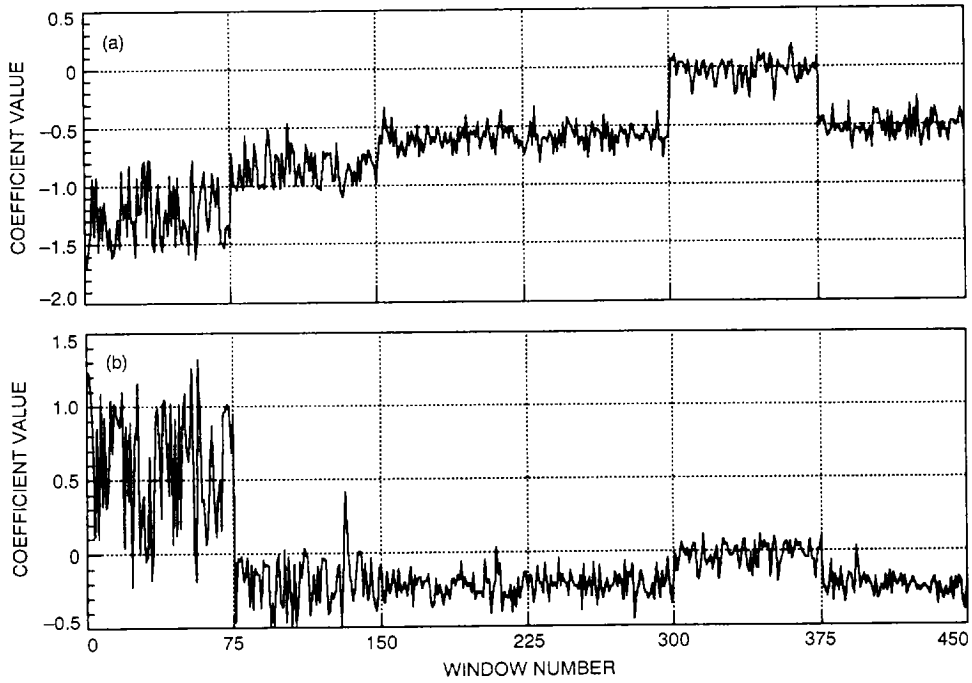


Fig. 6. Discrimination capabilities of (a) ARX coefficient 1, and (b) ARX coefficient 2 (day2 data).

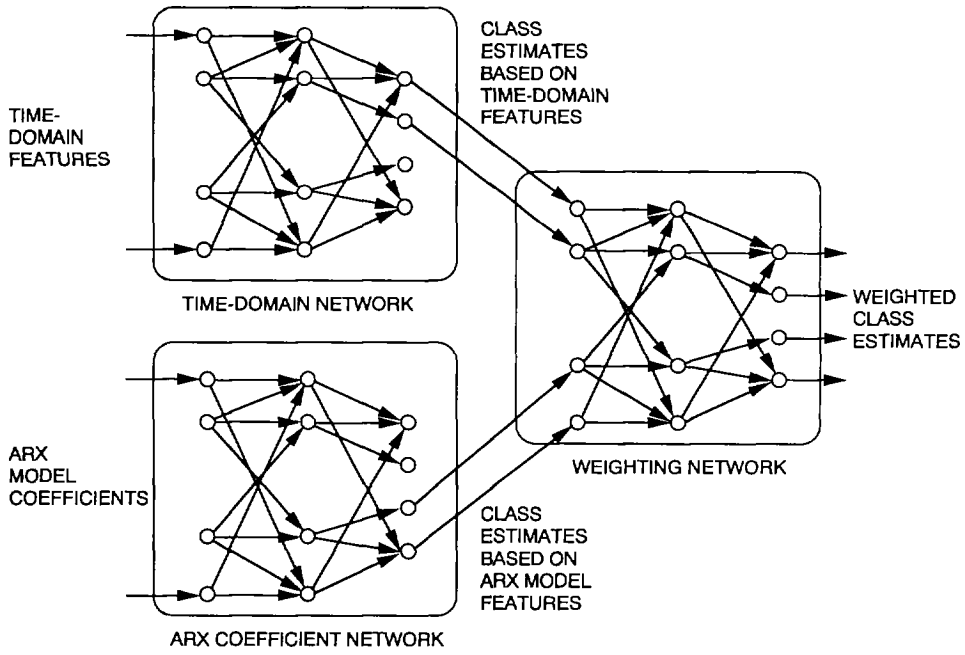


Fig. 7. Multiple-network architecture.

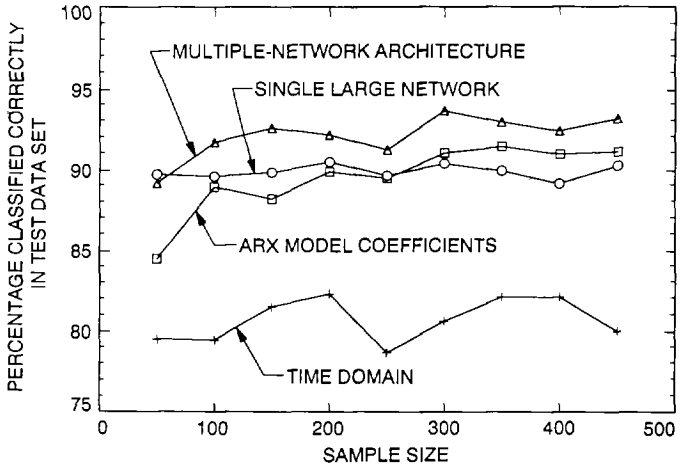


Fig. 8. Classification accuracy of classifier models as a function of sample size.

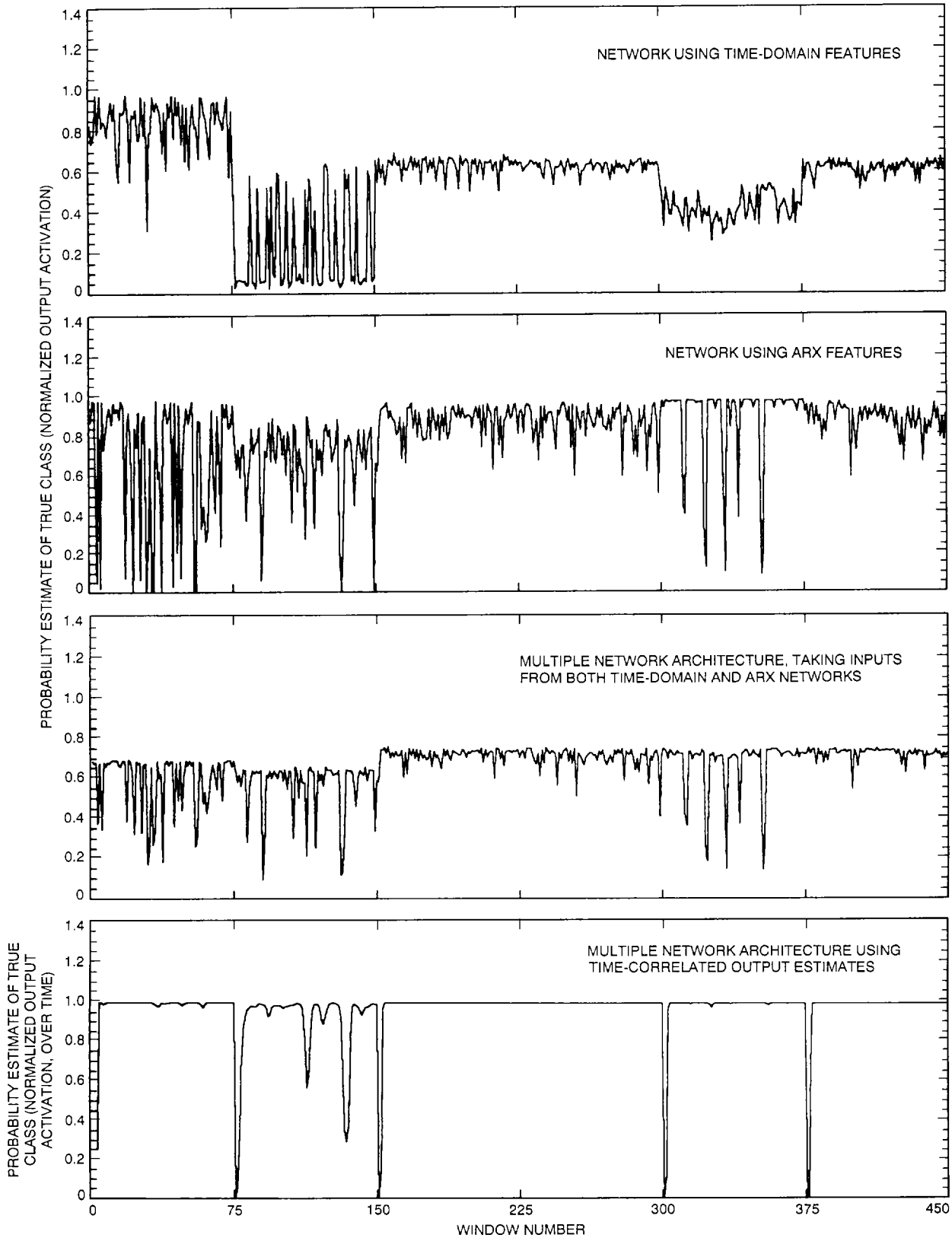


Fig. 9. Smoothing effect of using time-correlated information on a test data set (memory of size 5).

Appendix A

Neural Network Model Description

The following description of an example of a popular feed-forward multilayer neural network model will familiarize the reader with the general notation and concepts. This appendix is essentially a repetition of the Appendix in [3] and is included in order to make the report self-contained.

Figure A-1 shows an example of a network. The input nodes are labeled n_i, $1 \leq i \leq K+1$; the hidden nodes are labeled h_j, $1 \leq j \leq H$; and the output layers are labeled o_k, $1 \leq k \leq m$. In general, there are $K+1$ input units, where K is the number of features. The extra node is always in the "on" state, providing a threshold capability. Similarly, there are m output nodes, where m is the number of classes.

The number of hidden units H in the model is chosen based on rules of thumb and empirical experience. The size of this hidden layer can influence the classifier performance in the following manner: Too many hidden units, and the network overfits the data (i.e., the estimation error will be large); too few hidden units, and the network is left with insufficient representational power (i.e., the approximation error term is large).

Each input unit i is connected to each hidden unit j by a link with weight w_{ij}, and each hidden unit j is connected to each output unit k by a weighted link w_{jk}. Each hidden unit calculates a weighted sum and passes the result through a nonlinear function $F()$, i.e.,

$$a(h_j) = F\left(\sum_{i=1}^{i=K+1} w_{ij}\, a(n_i)\right)$$

where $a(n_i)$ is the activation of input unit i. Typically, this is just a linear (scaled) function of the input feature. A commonly used nonlinear function in the hidden unit nodes $F(x)$ is the so-called sigmoid function, defined as

$$F(x) = \frac{1}{1+e^{-x}}$$

Output unit k calculates a similar weighted sum using the weights w_{jk} between the jth hidden unit and the kth output unit, i.e.,

$$a_k = G\left(\sum_j w_{jk} a(h_j)\right)$$

where a_k is the activation of the kth output node. The function $G(x)$ can be chosen as either a linear (e.g., $G(x) = x$) or a nonlinear function. For example, for a classification problem such as that described in this article, the sigmoid function is used to restrict the range of the output activations to the range $[0, 1]$.

A classification decision is made by choosing the output unit with the largest activation for a given set of inputs (feature values); i.e., choose class k such that

$$k = \arg\max_i \{a_i\}$$

The network design problem is then to find the best set of weights such that a particular objective function is minimized on the N training data samples. The training data are in the form of input-output pairs $\{\underline{x}_i, y_i\}$, $1 \leq i \leq N$, where $\underline{x}_i$ is a feature vector and y_i is the desired output (for simplicity of notation, assume that there is just a single output model). Let $\hat{y}_i(\Omega, \underline{x}_i)$ be the network output for a particular set of weights Ω and input vector $\underline{x}_i$. The objective function is typically some metric on y_i and $\hat{y}_i$, whose mean value is estimated on the training data. Such commonly used objective functions include the mean-squared error

$$E_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i(\Omega, \underline{x}_i)\right)^2$$

and the cross-entropy error

$$E_{CE} = \frac{1}{N}\sum_{i=1}^{N} y_i \log\frac{y_i}{\hat{y}_i(\Omega, \underline{x}_i)} + (1-y_i)\log\frac{1-y_i}{1-\hat{y}_i(\Omega, \underline{x}_i)}$$

From a maximum-likelihood standpoint, the mean-squared-error approach assumes that the training data are perturbed by additive Gaussian noise, while the cross-entropy function assumes a multinomial distribution on

the class labels. Despite these differences, for classification problems there appears to be little significant difference between these objective functions; in fact, as shown in [5], these two error functions are the simplest functions in the general family of functions that can be proven to asymptotically produce consistent probability estimates. For the experiments reported in this article the mean-squared error objective function was used.

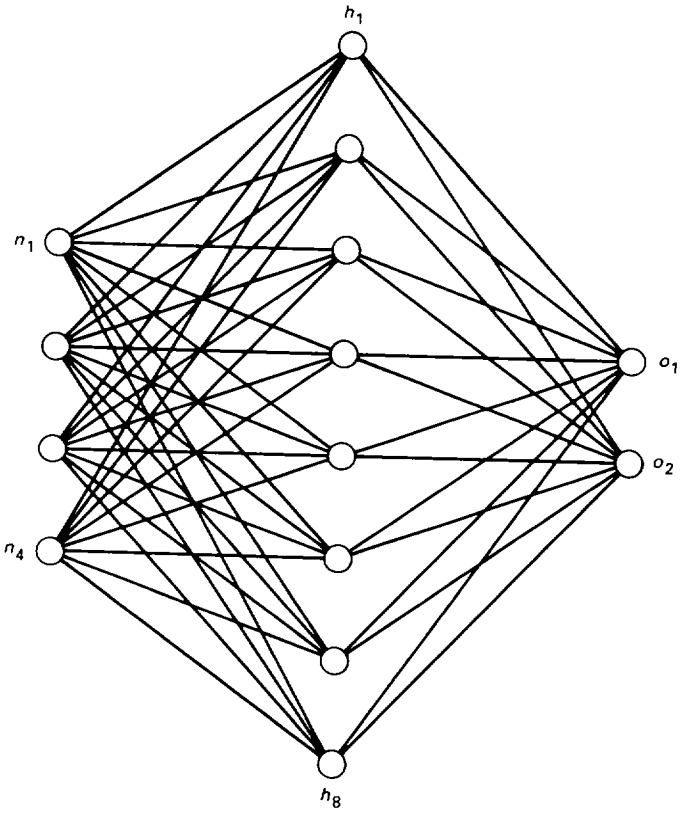


Fig. A-1. Example of a three-layer feed-forward neural network.

Appendix B

Error Probability as a Function of Memory Length

A simple model is proposed to determine the effects of memory length on probability of error when using the product of output activations over time as a classification metric. Let n be the number of terms in the product and $n = M + 1$, where M is the memory length referred to in the text. Assume a 2-class problem: this is a worst case since the multiclass case can be reduced to a 2-class problem where the noise is malicious in the sense that all misclassification errors occur in the form of one other particular class. Let o_1 be the classifier's output when class 1 is actually the true class, and let $\hat{o}_1$ be the output of the classifier when a noise hit occurs, i.e., the output for class 2 is larger than for class 1, even though class 1 is the true class. Such noise hits are likely, given the noisy environment in which the sensors operate. Outputs o_2 and $\hat{o}_2$ are defined in a similar manner; for example, one might have $o_1 = 0.8$, $o_2 = 0.1$, $\hat{o}_1 = 0.4$, and $\hat{o}_2 = 0.5$. Typically the network produces output activations where it is much more confident in its correct decisions than it is in its mistakes. Note that the simplifying assumption has been made that the outputs o_1, etc., are constant with respect to time. This makes the model simpler to analyze and gives a general idea of what is happening. A more sophisticated approach might be to assume that o_1 is the mean value of the activation which varies with some deviation σ_1.

In the case where class 1 is the true class, for a product of size n, if k noise hits occur, then the probability estimate for class 1 will only be greater than class 2 if

$$\sum_{n-k} \log \frac{1}{o_1} + \sum_{k} \log \frac{1}{\hat{o}_1} > \sum_{n-k} \log \frac{1}{o_2} + \sum_{k} \log \frac{1}{\hat{o}_2}$$

or

$$n > k(1 + \gamma)$$

where

$$\gamma = \log \frac{\hat{o}_2}{\hat{o}_1} / \log \frac{o_1}{o_2}$$

Hence the "error-correction" capability of a memory-based classifier depends on this parameter γ, which typically is quite small, perhaps about 0.1 in many cases, i.e., the classifier is 10 times more confident in its decisions when it is correct than when it is incorrect. A value of $\gamma = 1.0$ means that there is no difference, and the error-correction capability is effectively reduced to $n/2$ for a product of size n.

Now let p be the probability of such a noise hit. This parameter p measures the reliability of the classifier and the robustness of the features to withstand noise in the data. Assuming that these noise hits are not correlated (which may not be a realistic assumption in practice, but the model should be kept simple), then the error probability is given by the total number of ways in which at least $n/(1 + \gamma)$ errors can occur in a window of size n. This is simply the sum of binomial terms:

$$p_e = \sum_{i=\lceil \frac{n}{1+\gamma} \rceil}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

If $\gamma << 1$, $\lceil \frac{n}{1+\gamma} \rceil \approx n$ so that

$$p_e \approx p^n$$

i.e., the error falls off exponentially as a function of memory.